A Stochastic Game Model with Imperfect Information in Cyber Security

With the explosive growth of the Internet and its extensive use in all sectors, security has become a challenge as hackers are finding new ways to launch multistage attacks to cause damage to information assets. Despite considerable effort from the research community this problem is far from completely solved. Recently, researchers have started exploring the applicability of game theory to address this problem (Roy 2010). Since game theory deals with scenarios where multiple players with contradictory objectives compete with each other, it can provide a mathematical framework for analysis and modeling information system security challenges. The interaction between the attacks and the defense mechanisms can be considered as a game played between the attacker and the system administrator.

To model attacks and defense mechanisms a stochastic game model has been used in (Lye 2002), (Lye 2005) and (Baser 2006). The state of the game probabilistically changes depending on actions taken by the players (i.e., type of attacks and defender's response) and the system configurations. During each state transition, each player gets a payoff or incurs some cost (negative payoff). Techniques to determine the best strategy for a player, which will result in the highest overall payoff, considering all of the possible strategies of the adversary exist. Game theoreticians formulate the solution concept of a stochastic game by the notion of Nash equilibrium and have already provided the analysis indicating the existence of the equilibrium (Filar 1997).

However, the prior stochastic game models for network security (Lye 2002) (Lye 2005) assume that the players have perfect information about the current state of the game, which implies that the defender is always able to detect an attack and the attacker is always aware of the employed defense mechanism. In real systems, a player uses a sensor (e.g., the defender's sensor can be a part of the Intrusion Detection System (IDS)) to observe the current status of the system to decide the strategy. It is widely believed that no real sensor can perfectly read the environment, i.e., usually there is a non-zero error probability. So, in most cases, the above assumption about perfect information does not hold in real life.

Our work relaxes this assumption and designs a stochastic game model which is able to capture more realistic scenarios. It considers that a player knows the system's true state at a particular moment with some error probability, i.e., at any given point of time the true state and a player's perception can be potentially different. With this additional constraint of imperfect information, this work computes the best strategy for a player considering other player's choice of possible strategies.

In particular, our work presents a theoretical analysis by which the defender can compute his/her best strategy to reach the Nash equilibrium of a stochastic game assuming the defender's sensor is imperfect. It is implicit that the defender knows the error probability of his/her sensor and the players' objectives are directly opposite, i.e., it is a zero-sum game. Moreover, our work shows that if the defender follows the strategy prescribed by the perfect information model, then the Nash equilibrium is not achieved and the attacker's payoff can be more. Our algorithm for computing the best strategy runs offline well before the game is being played, i.e., our game analysis is static. Furthermore, our theoretical results are validated via simulation experiments in MATLAB.

The major contributions of our work are summarized below:

a. We present a static analysis of an imperfect information zero-sum stochastic game and compute the best strategy of the system administrator in realistic scenarios.

b. Our analysis and simulation experiments illustrate that the system administrator will be better-off if he/she takes our strategy compared to the scenario when he/she executes the strategy prescribed by the perfect information models.